システムソフトウェア・試験問題

2017 年度 (2017 年 11 月 27 日・試験時間 90 分)

書籍，配布資料およびノート等は参照してはならない．ただし，最大一枚までのメモ（手書きに限る．A4 両面使用可）を参照できるものとする．

1. xv6 のファイルシステムにおいて，inode ブロックに格納される `dinode` 構造体はソースコード 1 のように定義されている．

```
29 struct dinode {
30   short type;               // ファイルタイプ
31   short major;              // 主デバイス番号
32   short minor;              // 副デバイス番号
33   short nlink;              // リンク数
34   uint size;                // ファイルサイズ
35   uint addrs[NDIRECT+1];    // ブロック参照
36 };
```

ソースコード 1: fs.h

マクロ `NDIRECT` は 12 と定義されている．`addrs[0]` から `addrs[NDIRECT-1]` の 12 個がデータブロックへの直接参照で，`addrs[NDIRECT]` が間接参照である．またブロックサイズは 512 バイトである．

(a) xv6 では最大何バイトまでのファイルを扱うことができるか．ただしディスクは十分大きく，ディスクサイズによる制約はないものとする．

(b) 10000 バイトのファイルが占めるデータブロックの数はいくつか．間接参照ブロックが必要な場合はそれも数えること．i-node，ビットマップ，ログのためのブロックは数えなくてもよい．

(c) システムコール `unlink` は引数で指定されたファイルを消去したいときに用いられる．その動作としては，まず引数に該当する inode のフィールド `nlink` の値を 1 減らし，そして `nlink` が 0 になった場合に inode およびファイルが使用していたブロックの解放を行う．このように `nlink` が 0 になるまで実際の消去を行わない理由を述べよ．

2. 4 つの物理ページを持つシステムがある．ページ p の参照ビットを R_p，汚れビットを D_p とし，それぞれ当該ページの読み出しと書き込みが行われる際にセットされるものとする．時刻の下 2 桁が 0 のとき，各ページの参照ビットはクリアされる (汚れビットはそのまま)．次にページのクラスを次のように定める．$R_p = 0, D_p = 0$ のとき p はクラス 0 とし，同様に $R_p = 0, D_p = 1$ のときクラス 1，$R_p = 1, D_p = 0$ のときクラス 2，$R_p = 1, D_p = 1$ のときクラス 3 とする．ページフォルトが起きた時点で，クラスが一番小さいページから一つ選んで犠牲ページとするアルゴリズムを考える (NRU アルゴリズムと呼ばれる)．

いま，時刻 1000 においてページ 0〜3 がロードされ，それぞれの参照ビットと汚れビットが 0 にされたとする．その後以下のようなアクセス系列 (R は読み出し，W は書き込みを表す）があったとする．

	ページ (p)			
時刻	0	1	2	3
1020	W			
1045				R
1130				W
1195		R		
1235			R	
1305				R
1335	R			
1355		R		
1435			R	
1445	R			

(a) 時刻 1320 における各ページのクラスを答えよ．

(b) 時刻 1450 において犠牲ページを選ぶ必要が生じたとする．このときの犠牲ページはどれか．

(c) そのときページの書き戻しは必要か．理由も簡単に説明せよ．

3. ソースコード 2〜4 は xv6 のコンテクストスイッチングに関連する部分である．これらを用いてソースコード 5 で示されるプログラムを書いた．ここで `uint` は **`typedef`** によって **`unsigned int`** と定義されているものとする．また **`int`** 型およびポインタ型のサイズは 32 ビット (4 バイト) とする．

(a) このプログラムを実行したところ，1 番目に `A`，2 番目に `C` が出力された．3 番目，4 番目，6 番目および最後から 2 番目に出力された文字はそれぞれ何か．

(b) このプログラムが停止するまでに全部で何文字出力されるか．改行文字はカウントしないものとする．

```
8  .globl swtch
9  swtch:
10   movl 4(%esp), %eax    # 第1引数
11   movl 8(%esp), %edx    # 第2引数
12
13   # Save old callee-save registers
14   pushl %ebp
15   pushl %ebx
16   pushl %esi
17   pushl %edi
18
19   # Switch stacks
20   movl %esp, (%eax)
21   movl %edx, %esp
22
23   # Load new callee-save registers
24   popl %edi
25   popl %esi
26   popl %ebx
27   popl %ebp
28   ret
```

ソースコード 2: swtch.S

```
27 struct context {
28   uint edi;
29   uint esi;
30   uint ebx;
31   uint ebp;
32   uint eip;
33 };
```

ソースコード 3: proc.h

```
125 void swtch(struct context**, struct context*);
```

ソースコード 4: defs.h

(c) このプログラムを逆アセンブルしたところ，関数 `foo` の最初の呼び出しには `call` 命令が使われていたが，関数 `bar` の最初の呼び出しは別の命令の実行によっていた．その命令を答えよ．

(d) ソースコード 5 の 11 行目における `i < 4` の部分を `i < 5` に変更して実行したところ，プログラムはエラーで停止した．その理由を答えよ．

(e) xv6 ではタイマ割込みによってカーネル内で `swtch` を用いており，そのためユーザプログラムでは明示的なコンテクストスイッチングを行う必要はない．このような仕組みにもとづくプロセススケジューリング方式を何と呼ぶか．

```
7  #define STACK_SIZE 4096
8  struct context *foo_ctx, *bar_ctx;
9
10 void foo() {
11     for (int i = 0; i < 4; i++) {
12         printf("A\n");
13         swtch(&foo_ctx, bar_ctx);
14         printf("B\n");
15     }
16 }
17
18 void bar() {
19     for (int i = 0; i < 2; i++) {
20         printf("C\n");
21         swtch(&bar_ctx, foo_ctx);
22         printf("D\n");
23         swtch(&bar_ctx, foo_ctx);
24         printf("E\n");
25     }
26 }
27
28 int main() {
29     uint *sp = malloc(STACK_SIZE);
30     sp += STACK_SIZE / sizeof(uint);
31     *--sp = 0;
32     sp -= sizeof(struct context) / sizeof(uint);
33     bar_ctx = (struct context *)sp;
34     bar_ctx->eip = (uint)bar;
35     foo();
36     return 0;
37 }
```

ソースコード 5: ex.c